**StepBox Bluetooth communication protocol**

**1) Overview**

- Implemented using ArduinoBluetoothAPI assets

[For asset usage, refer to BluetoothAPI/Android Unity Plugin.pdf]

- Can transmit/receive data of variable length using Length Based Stream

(See page 6 of Android Unity Plugin.pdf)

Example) When sending {0x02, 0x04, 0x65, 0xE5} data

Transmitted in the form {0x55, 0x55, 0x00, 0x04, 0x02, 0x04, 0x65, 0xE5}.

- {0x55, 0x55} is a preamble bit synchronization signal. [0b01010101]

- {0x00, 0x04} is the actual length of the data ( {0x00, 0x04} is 4 bytes)

- {0x02, 0x04, 0x65, 0xE5} is the actual data

**2) Step-up communication packet**

- Bluetooth packet structure (6 bytes)

\* 0 ~ 3: Player ID (int 4 bytes)

\* 4: Sensor ID (byte 1 byte)

\* 5: Sensor status (byte 1 byte)

- Player ID determines which step-up device is used (default = 0)

- The sensor ID is the sensor number [ex. left=0, right=1]

- The sensor state determines whether the sensor is pressed or released (released=0x00, pressed=0xFF)

- When do you send a signal?

- Only the moment when the sensor is pressed or released

**3) Related scripts**

- BluetoothInputHandler - Handles rhythm game notes according to Bluetooth sensor input

(Inherits the ISensorInputHandler interface)

<<<<<<<<<<<>>>>>>>>>>>>>>>

**StepBox Server API Cleanup**

**login.php - Attempt** to  **log in** **to the server**

**POST parameters**

|  |  |
| --- | --- |
| field | details |
| Id | User ID |
| password | User password (plain text) |

**Result: JSON**

|  |  |
| --- | --- |
| 속성 Property | Details |
| Result | Success : Fail |
| Session Id | Session ID on successful login |
| Message | Reason for login failure: ID mismatch: ID does not exit  Password \_mismatch: password do not match  Session error: session error |

**check login.php - Check if you are logged in**

**POST parameters**

|  |  |
| --- | --- |
| field | detail |
| Session ID | Session ID |

**Result: JSON**

|  |  |
| --- | --- |
| 속성 Property | detail |
| Result | Login true or false |

**logout.php – logout**

**POST parameter**

|  |  |
| --- | --- |
| field | detail |
| Session ID | Session ID |

**Result JSON**

|  |  |
| --- | --- |
| 속성 Property | Detail |
| Result | Success: success, Failure: fail |

**register.php – Register**

**POST Parameter**

|  |  |
| --- | --- |
| field | details |
| Id | ID |
| Password | Password (plain text) |
| Name | name |
| Birthday | DOB 19901224 |
| Phone | 01069490699 |
| welfare | Affiliated Welfare Center Information |

**Result: JSON**

|  |  |
| --- | --- |
| Property 속성 | detail |
| result | Success: success, Failure: fail |
| error | Field error array  id\_error: id format or length error  pw\_error: password format or length error  name\_error: name error  birthday\_error: date of birth error  phone\_error: phone number error |
| Session ID | Automatic login session ID after successful membership registration |

**Check duplicateid.php - Check for duplicate IDs**

**POST parameters**

|  |  |
| --- | --- |
| Field | details |
| Id | IDs to check for duplicates |

**Result: JSON**

|  |  |
| --- | --- |
| 속성 Property | detail |
| Result | Duplicate: true, non-duplicate: false |

**fetchUserBasic.php - fetch user's basic information**

**POST parameters**

|  |  |
| --- | --- |
| Field | Details |
| ID | User ID |

**Result: JSON**

|  |  |
| --- | --- |
| Field | Details |
| Name | username |
| Welfare | Welfare center |

Things to hand over from 2021.02.17

**recordScore.php - Record score based on current time**

**POST parameters**

|  |  |
| --- | --- |
| Field | Details |
| Session ID | Login session ID |
| Content ID | Content ID |
| Score | score (real) |
| attempts | The number of attempts of the content per day |

**Result: JSON**

|  |  |
| --- | --- |
| 속성 Property | Details |
| Result | Success: success, Failure: fail |

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

**Application interface**

**Step-up core screen configuration**

**1. Start Screen (StartScene)**

- Splash screen

- Get the patch list from the server, check if it needs to be updated, and if it is, download the necessary files from the server and update the content.

**2. Main Screen (MainMenuScene)**

- Main menu screen

- There are 3 buttons

(1) “Start!” - Go to LoginScene

(2) “Settings” - Summon SettingsPanel

(3) “Exit” - close the app

- When running on Windows, the ‘Editor’ button is visible (Developer function)

**3. Login Screen (LoginScene)**

- ID and password input fields, “Login” and “Membership” buttons exist

- When the “Login” button is clicked, a POST is sent to the server and attempts to log in

- If the login is successful, move to the member information screen (InfoScene)

- If it fails, the error message (MessageText) shows the cause.

- Click the “Register” button to move to the RegisterScene screen.

**4. Member registration screen (RegisterScene)**

- Enter personal information (name, date of birth, phone number, welfare center, etc.) and login information (ID, password) into the fields

- Responsive UI design: When one input field is finished, it immediately validates and informs the result with the field background color or text.

- All fields must be valid for the “Join” button to be activated.

- If you click the “Join” button at the bottom, information is sent to the server and actually registered in the member DB.

- When member registration is successful, move to the member information screen (InfoScene)

**5. Member information screen (InfoScene)**

- Displays logged-in member information (progress, exercise tips, ranking, etc.)

- "start!" Click the button to go to the game start screen (AfterLoginScene)

**6. Game start screen (AfterLoginScene)**

- There are two options.

- “Continue”: Continue the next content of the previous content

- “Choose exercise”: Select one of all content to play

- Click the “Continue” button to go directly to the game screen (GameScene)

- Click the “Select Exercise” button to move to the content selection screen (LevelSelectScene)

**7. Content selection screen (LevelSelectScene)**

- You can play the practice level and the actual level separately by selecting the desired level.

- Click the “Practice” button to play both practice and practice levels

- If you click the “Start Play” button, you can only play the actual level

- Go to the game screen (GameScene) when you click the “Practice” or “Start practice” buttons

**8. Game Scene (GameScene)**

- It consists of a layout that displays exercise videos on the full screen, notes on the right, and motion actions on the left.

- Logic to take notes coming down with your feet on the stepper (Bluetooth) at the correct timing.

- Press the pause button to pause the game.

- Pause menu

- “Continue”: Resume game play.

- “Restart”: restart the game

- “Preferences”: Summon Settings Panel

- “Go to the main menu”: literally go to the main menu

- When the game is over, the result window is displayed. In the result window, there are score, count of each judgment, and 2 option buttons.

- Options in the result window

- “Continue”: Skip to the next content

- “Select an exercise”: Go to the content selection screen (LevelSelectScene).

**9. EditorScene**

- Editor to edit the transcript of content

- You can edit the transcript file by calling it, or create a new transcript and save it as a file.

- You can write transcriptions in the form of a timeline.

- Please refer to the “Editor Guide” for how to use the editor.

<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

**Specification of Transcript**

**1. File Format: JSON**

**2. Structure: example**

**2-1. “settings”: Includes song & video path, information, BPM, sync offset, etc. required for play.**

|  |  |
| --- | --- |
| Field | Details |
| Artist Name | Singer (composer) name |
| Music Name | Song (Music) Title |
| Music Path | Relative path to song (music) audio file |
| Video Path | Relative path to video file |
| Bpm | Initial BPM of the song (music) |
| Offset | Song (music) start offset (unit: seconds) |
| Speed | the speed at which notes fall |
| Pitch | Song (music) playback speed (default=1.0) |
| Video Sync | Start offset of the video (in seconds) |

**2-2. “notes”: arrangement of notes to be displayed in game play and included in verdicts- note object**

|  |  |
| --- | --- |
| Field | Details |
| Offset | Quarter note beat |
| Track | what line is the note |

**2-3. “events”: timeline events other than notes that have a visual indication or direct or indirect effect on gameplay**

|  |  |
| --- | --- |
| Filed | Details |
| Offset | Quarter-note time signature (determining when it will play) |
| Track | [Editor only] Track number to be placed |
| Type | Event type (string) |
| (additional data) | Data according to different types of events |

**2-3-1. BPM Change Event (BpmChangeEvent)**

|  |  |
| --- | --- |
| Field | Detail |
| bpm | BPM (float) to be changed immediately after the event |

**2-3-2. Note speed change event (SpeedChangeEvent)**

|  |  |
| --- | --- |
| Filed | Detail |
| Speed | The speed at which the note descends immediately after the event (float) |

**2-3-3. Stop Event (StopEvent)**

|  |  |
| --- | --- |
| Filed | Detail |
| Duration | The period during which notes stop falling from immediately after the event  (beat, float) |

**2-3-4. Text display event (ShowTextEvent)**

|  |  |
| --- | --- |
| Field | Details |
| Duration | How long the text is displayed from immediately after the event (beat, float) |
| Text | Text to display (string) |
| Color | Text color (hexadecimal) ex) FFFFFF - White |

**2-3-5. Section Repeat Event (RepeatEvent)**

**- Section repeat events are only used in the practice level, and cannot be used in the actual level.**

|  |  |
| --- | --- |
| Field | Details |
| Tag | Identifier tag of repeat interval |
| Duration | Length of the repetition period (in beats)  -> Interval start: offset, end: (offset + duration) |
| Description | Description of the interval (string) |

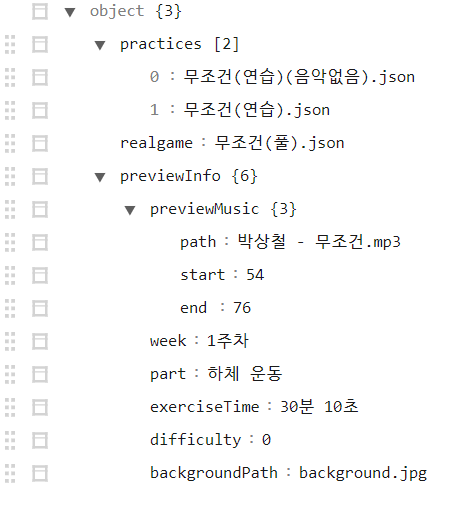
**2-3-6. Motion Display Event (MotionChangeEvent)**

|  |  |
| --- | --- |
| Filed | Details |
| Duration | BPM (float) to be newly applied immediately after the event |
| Motion Sequence | motion animation frame array (string)  Format: [Frame1] [Frame2] [Frame3]  - Frame: [Motion No.]:[Length (beat)]  ex) 11:1 9:1 10:1 13:1 8:1 5:1 7:1 13:1 |
| Motion Speed | Motion playback speed (default: 1.0) |
| IsLoop | Whether to repeat playback |
| Name | motion display name |

**<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>**

**Content Package File Specification**

**1. File name: playinfo.json**

**2. File Format: JSON**

**3. Structure: Example**

**3-1. “practices”: list of relative paths to practice level (transcript) files (array)**

**3-2. “realgame”: the relative path of the actual level (transcript) file (string)**

**3-3. “previewInfo”: required data (object) to be displayed in the exercise selection menu**

**\*** previewMusic: preview music (path, music playback section)

- path: relative path of preview music

- start: start of music playback section (seconds)

- end: End of music playback section (seconds)

\* week: week n

\* part: exercise part

\* exerciseTime: exercise time (still a string)

\* difficulty: exercise difficulty

\* backgroundPath: the relative path of the thumbnail image

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

**Core Script**

**Key Script Features Summary**

**(1) game control**

[RhythmGame/Core folder]

TimeManager - Manage in-game time (to sync with audio)

RhythmGame - responsible for loading game levels, starting and pausing the game

GameController - Interact with RhythmGame by loading transcriptions needed for the current play before the game starts

HitsoundPlayer - Plays clap and other sound effects when the notes are correctly timing

**(2) level (chaebo) container**

[RhythmGame/Core folder]

TimingData - Stores and manages all events in the level timeline and calculates time and note location

LevelData - Contains the data needed for the level

LevelLoader - Parsing level JSON files

**(3) Timeline events**

[RhythmGame/Events folder]

**GameEvent** - A basic abstract unit of action that occurs in a sequence, such as notes, shifts, repetitions, and motion.

NoteEvent : Rhythm game note

BpmChangeEvent : BPM shift

SpeedChangeEvent: change the note fall speed

StopEvent : Stops the note dropping for a certain period of time (beat).

ShowTextEvent : Show in-game text

RepeatEvent : Section repeat

TeleportTimeEvent : Forced movement of music and video time

PauseEvent: Force the game to pause

MotionChangeEvent : Show motion motion

**EventProcessor** - Detect and process specific types of events in-game in real time

ShowTextEventProcessor - text display event handling (show, fade in & out)

PauseEventProcessor - game pause event handler (+unpause)

TeleportTimeEventProcessor - Music and video time force event handler

MotionChangeEventProcessor - kinematic motion display event handler (show, fade in&out)

**(4) notes**

[RhythmGame/Notes folder]

INoteObject - the main interface for notes

NoteObject - the actual note associated with the NoteEvent

NoteGenerator - Creates a Note GameObject by pulling it out of the object pool when the note position is below the upper limit

NoteAppearenceSettings - ScriptableObject that stores the appearance of notes

**(5) Judgment**

**[RhythmGame/Core/Judgements folder]**

NoteJudgement - Judging n button notes with NoteLineJudger via input interface and consolidating the results

NoteLineJudger - Takes input of 1 individual button and judges notes on a specific line -> Individual line note judger

JudgmentType - enumeration of judgment types (Perfect, Good, Bad, Miss, None)

JudgmentInfo - Contains information about judgment (type, note event, time difference)

JudgmentTiming - store judgment timing as a static constant

**(6) input**

[Input folder]

ISensorInputHandler - Default interface for sensor input (press, release detection)

KeyboardInputHandler - keyboard input interface

BluetoothInputHandler - Bluetooth (Arduino) sensor input interface

**(7) video**

[Game folder]

StreamVideo - Interact with RhythmGame to load and play video

**(8) Content**

StepupContent - Contains [practice level + practice level] and content information (difficulty level, song name, parking).

ContentManager - Responsible for loading and accessing contents locally

**\* Libraries used:**

Newtonsoft.Json - Parsing JSON

ArduinoBluetoothAPI - Bluetooth pairing & connection and sending and receiving sensor data

DOTween - UI Tween (Animation)